Cut, Copy, and Paste on the iPhone

Session 105

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

Cut, Copy, and Paste on the iPhone

Andrew Platzer
iPhone Frameworks Engineer

Editing and Sharing Your Data

- Display the edit control
- Specify which actions are available
- Share your data using the pasteboard
- Add undo to your application

New Features

- Menu controller
- Responders
- Pasteboard
- Undo manager

Menu Controller

Menu Controller

- Provides access to edit interface
- Single shared instance
- Position and show the control
- Enable items

Showing the Menu

- Use -setMenuVisible:animated:
- Trigger on ‘press-and-hold’ gesture

Positioning the Menu

- Use -setTargetRect:inView:
- targetRect can be zero width or height
- Doesn’t track changes to view location
- Set while visible to move

Example

```
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event {
    for (UITouch *touch in touches) {
      if ([self pointInside:[touch locationInView:self] withEvent:event]) {
        [self becomeFirstResponder];
        if (event.timestamp > beganTimestamp + HOLD_TIME) {
            UIMenuController *controller = [UIMenuController sharedController];
            [controller setTargetRect:self.bounds inView:self];
            [controller setMenuVisible:YES animated:YES];
        }
        break;
      }
    }
}
```

Hiding the Menu

- Automatically hidden
  - Touch begins in another view
  - Alert appears
  - Application deactivates
- Hide via -setMenuVisible:animated: or UIMenuController.menuVisible
- Listen for UIMenuControllerWillHideMenuNotification

Menu Interface

- Automatically localized
- *Not* customizable
  - *Fixed* item names
  - *No* additional items
  - *Don't* even try

Responders

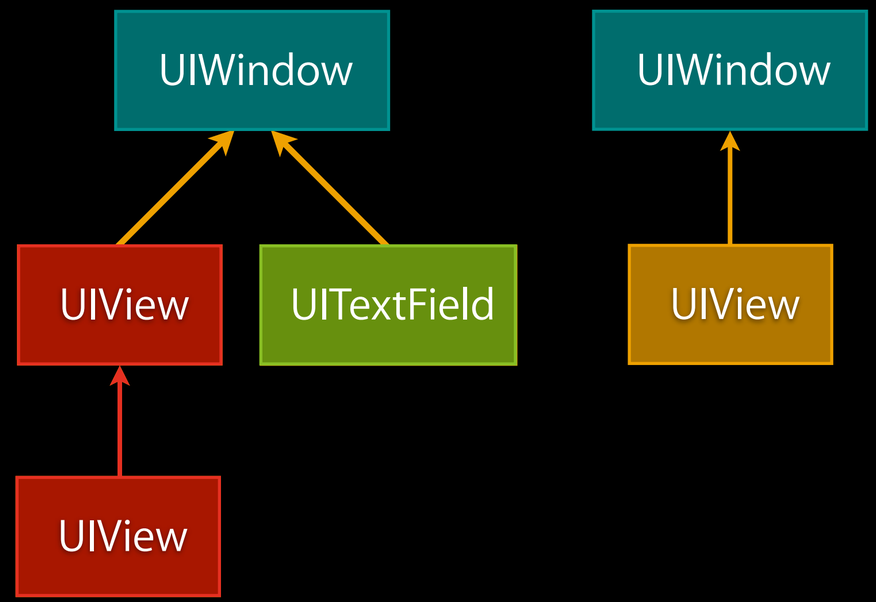
Responders

In 2.x

- Only for passing touch events from view to superview
- UIApplication.keyWindow not used
- Only UITextField and UITextView became first responder

Responders

In 2.x



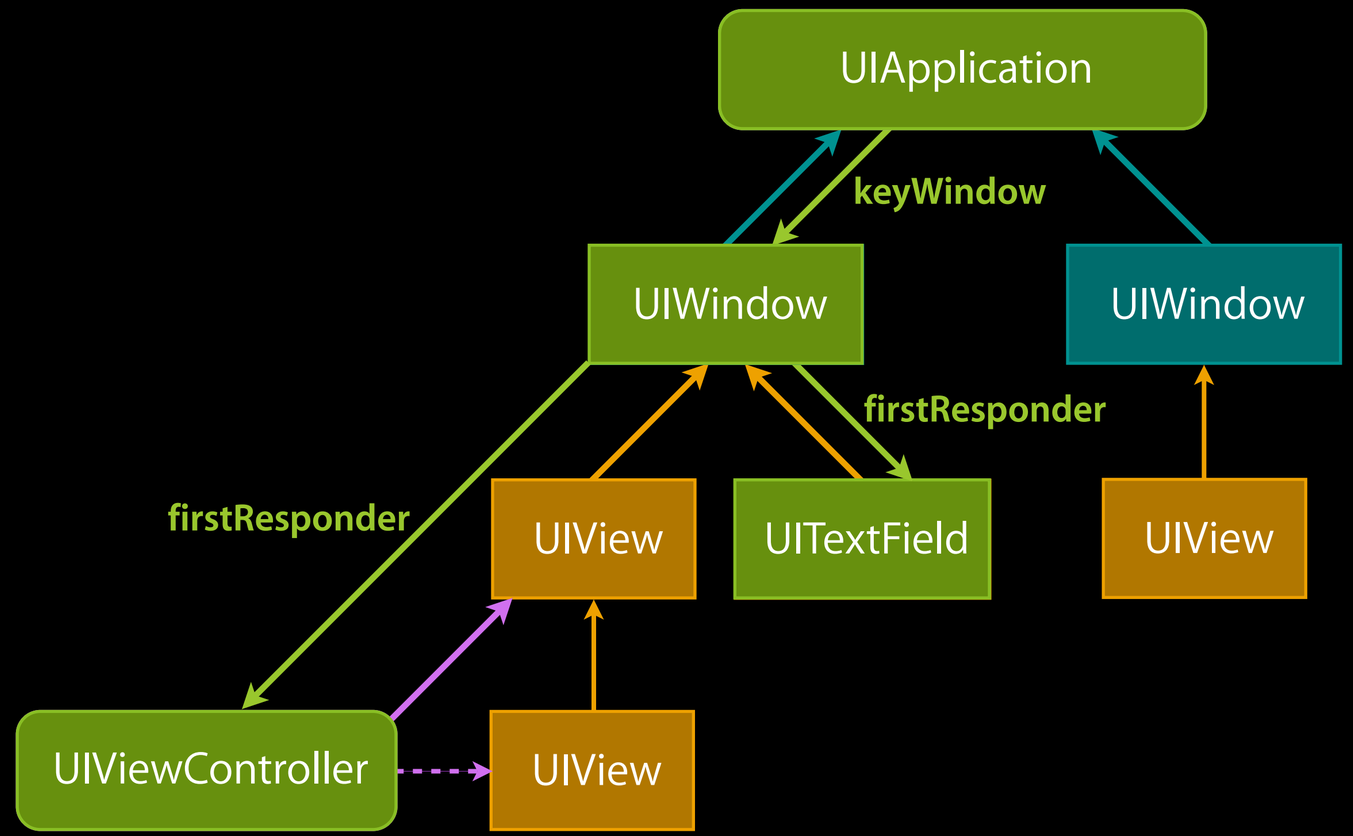
Responders

In 3.0

- UIApplication.keyWindow is used
  - -[UIWindow makeKeyAndVisible]
  - UIWindow.keyWindow
- Used for menu action validation and handling
- Be key window and first responder aware

Responders

In 3.0



Responders

- UIView, UIViewController and UIApplication are UIResponders
- Chained using -nextResponder
  - Read-only property
  - Subclass to return custom next responder
- Each UIWindow has a first responder

Becoming the First Responder

- -canBecomeFirstResponder
  - Read-only property defaults to NO
  - Subclass to return YES
- Call -becomeFirstResponder
- Override -becomeFirstResponder and -resignFirstResponder to show feedback
- Must be in a window to become the first responder

Becoming the First Responder

- With a view

```
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event {
    for (UITouch *touch in touches) {
      if ([self pointInside:[touch locationInView:self] withEvent:event]) {
        [self becomeFirstResponder];
        ...
        break;
      }
    }
}
```

Becoming the First Responder

- With a view controller

```
- (void)viewDidAppear:(BOOL)animated {
  [super viewDidAppear:animated];
  [self becomeFirstResponder];
}
```

Responders Coming and Going

- UITextField and UITextView
  - Become first responder on touch
- Listen for resigning first responder using
  - -textFieldDidEndEditing:
  - UITextFieldTextDidEndEditingNotification

Responder Actions

- Predefined system editing actions

```
- (void)cut:(id)sender;
- (void)copy:(id)sender;
- (void)paste:(id)sender;
- (void)select:(id)sender;
- (void)selectAll:(id)sender;
```

Enabling Actions

- Implementing the action
  - Always enabled
  - Standard actions are not implemented by UIResponder
- Add -[UIResponder canPerformAction:withSender:]
  - Customize when the action is available
  - Called for every action
  - Always pass unknown actions to super

Enabling Actions

```
- (void)canPerformAction:(SEL)action withSender:(id)sender {
  if (action == @selector(cut:)) {
    return NO;
  } else if (action == @selector(copy:)) {
    return self.selectedCount > 0;
  } else if (action == @selector(paste:)) {
    return NO;
  } else if (action == @selector(select:)) {
    return NO;
  } else if (action == @selector(selectAll:)) {
    return self.itemCount > 0;
  } else {
    return [super canPerformAction:action withSender:sender];
  }
}
```

Demo

Pasteboard

Pasteboard

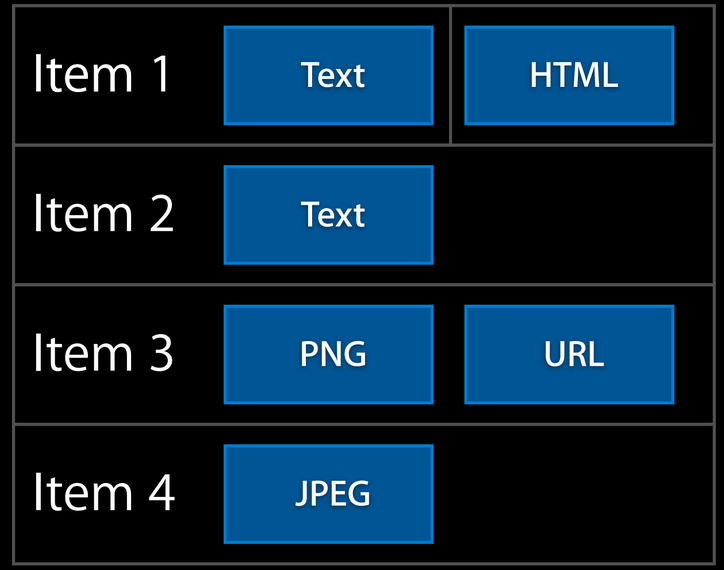
- Shared and persistent storage
- Single or multiple items
- Multiple representations per item
- Standard pasteboard

Getting a Pasteboard

- Get the system pasteboard
  - +generalPasteboard
- Get or create named pasteboards
  - +pasteboardWithName:create:
  - +pasteboardWithUniqueName
- Pasteboards are owned by the application that created them

Pasteboard Items

- Single item
- Multiple, unique representations per item
- Multiple items



Pasteboard Data

- All information stored as NSData
  - -dataForPasteboardType:
- Can use property list objects
  - -valueForPasteboardType:
    - NSDictionary, NSArray, NSDate, NSString, NSNumber, NSData

Pasteboard Contents

- Check if type is on pasteboard
  - -containsPasteboardTypes:
- Or ask for a list if order is important
  - -pasteboardTypes

Single Pasteboard Item Methods

- Operates on first item only

```
- (NSArray *)pasteboardTypes
- (BOOL)containsPasteboardTypes:
- (NSData *)dataForPasteboardType:
- (void)setData:forPasteboardType:
- (id)valueForPasteboardType:
- (void)setValue:forPasteboardType:
```

- Setting will replace all items

Multiple Items

- Same methods as single item
  - Adds UIPasteboard.numberOfItems
- Use NSIndexSet to specify items of interest
- Returns array instead of single item

```
-(NSData *)dataForPasteboardType:(NSString*)pbType
-(NSArray *)dataForPasteboardType:(NSString *)pbType inItemSet:(NSIndexSet *)set;
```

- Direct access via UIPasteboard.items and -[UIPasteboard addItems:]
  - UIPasteboard.items returns everything
  - Can use multiple items methods after setting single item

Persistence

- Application pasteboards are not persistent
  - Removed when application exits
- Setting persistence flag will keep pasteboard around
  - Do not use as general storage
  - Not guaranteed to be around after reboot
- Owner can call +removePasteboardWithName:

Predefined Pasteboard Types

- Strings, images, URLs, colors

```
[UIPasteboard generalPasteboard].string = @"Hello, World!";
```

- Single and multiple item properties

```
[UIPasteboard generalPasteboard].strings
          = [NSArray arrayWithObjects:@"Hello", @"World!", nil];
```

- Setter clears all existing items
- Images are converted to PNG
  - Use original image data if possible
  - Can be noticeably slow

UTIs

- Use UTIs (“Universal Type Identifiers”)
  - kUTTypePNG, kUTTypeMP3, kUTTypeURL
- UIPasteboard type lists
  - UIPasteboardTypeListImage = kUTTypePNG, kUTTypeTIFF, kUTTypeJPEG, kUTTypeGIF
- Same API as Mac OS X
- Located in MobileCoreServices.framework

Pasteboard Do's and Don'ts

- Do add multiple representations
- Don't use for storage
- Do use original image data
- Don't put secure data on the general pasteboard

Demo

Undo

Undo

- Mac OS X NSUndoManager
- Register undo action when user makes a change
- The undo manager
  - Keeps stack of actions
  - Plays back actions
  - Group actions together
  - Keeps track of action name

Creating an Undo Action

- With target/selector
  - -[NSUndoManager registerUndoWithTarget:selector:object:]
- With invocation
  - -[NSUndoManager prepareWithInvocationTarget:]

Undo Action with Target/Selector

- Use target/action with
  - Reversible setter
  - Undo method that takes a single NSObject

```
- (void)setColor:(UIColor *)color {
    if (![color isEqual:_color]) {
        NSUndoManager *undoManager = [self undoManager];
        [undoManager registerUndoWithTarget:self
                                   selector:@selector(setColor:)
                                     object:_color];
        [_color release];
        _color = [color retain];
        [self update]
    }
}
```

Undo Action with Invocation

- Use invocation with
  - Non-NSObject parameters
  - Multiple parameters

```
- (void)setLevel:(NSInteger)level {
    if (level != _level) {
        NSUndoManager *undoManager = [self undoManager];
        [[undoManager prepareWithInvocationTarget:self] setLevel:_level];
        _level = level;
        [self update];
    }
}
```

Undo Title

- Set the name when you register the action

```
- (void)setColor:(UIColor *)color {
    if (![color isEqual:_color]) {
        NSUndoManager *undoManager = [self undoManager];
        [undoManager registerUndoWithTarget:self
                                   selector:@selector(setColor:)
                                     object:_color];
        [undoManager setActionName:@"Set Color"];
        …
    }
}
```

Managing Undo State

- Nesting undo actions
  - Default is to group for a single event
  - Combine same actions across events
    - -beginUndoGrouping/-endUndoGrouping
- Clear undo stack
  - -removeAllActions
  - -removeAllActionsForTarget:

Enabling Undo

- Automatically get ‘shake to undo’
  - Alert shows only if there is something to undo
- Turn off via property
  UIApplication.applicationSupportsShakeToEdit
- Always get motion events
- UIApplication must receive motion events to enable undo

Undo Managers

- Shared undo managers
  - Searches up responder chain
  - Each window has one
- Special undo managers
  - UITextField
  - NSManagedObjectContext in CoreData
- Provide your own for ‘modal’ undo

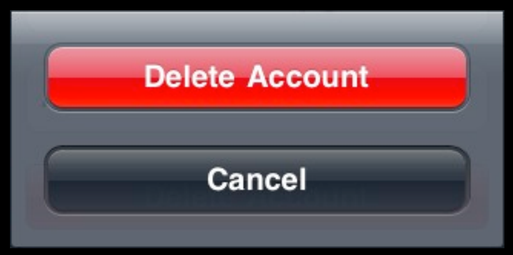
Undo Interface

- Undo should have visible effect
  - Don’t undo offstage
  - Show selection
  - Use animation
- Clear undo stack when leaving
- Undo and Redo toolbar buttons

When Not to Undo

- Don’t use undo for large scale actions
- Use a two stage confirmation
  - Minus button + delete button in table view

  - Delete button + action sheet



Demo

More Information

Matt Drance
Application Frameworks Evangelist
mdrance@apple.com

Documentation
UIKit Reference
http://developer.apple.com/iphone

Related Sessions

Effective iPhone App Architecture	Presidio Thursday 3:30PM
iPhone View Controller Techniques	Presidio Thursday 5:00PM
What's New in Cocoa Touch	Pacific Heights Tuesday 10:30AM
iPhone Views and Animations	Pacific Heights Tuesday 3:30PM

Labs

iPhone Cut, Copy, and Paste Lab	iPhone Lab C Wednesday 5:00PM
iPhone View Controllers Lab	iPhone Lab A Friday 9:00AM

Summary

- Cut, Copy, and Paste let your users edit data
- The pasteboard lets your users share data
- Undo helps your users fix their mistakes
- Enhances your application's value

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09